

ONLINE COFFEE SHOP DATABASE DESIGN

A detailed design document for building a coffee
shop ecommerce database logic.

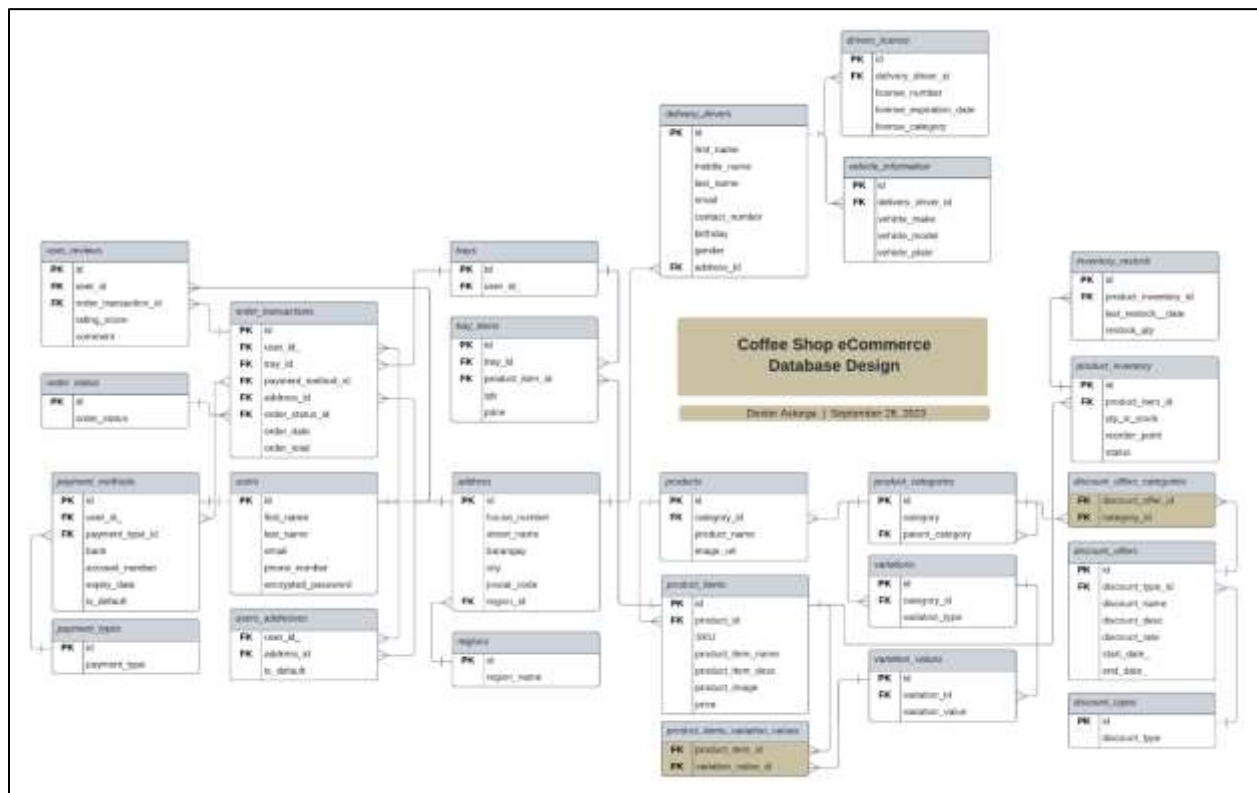
Dexter Astorga

<dev>
</dev>

This document contains:

- Entity-Relationship Diagram (ERD)
- Table Descriptions
- SQL Scripts (for creating the database tables)
- Sample data (each table)
- Example queries
- Conclusion

ENTITY-RELATIONSHIP DIAGRAM



This diagram is available here: https://github.com/Dex-Astorga/coffee-shop-ecom/blob/main/online_store_coffee_shop.png?raw=true

<dev>
</dev>



TABLE DESCRIPTIONS

users

Holds all account users

column	description
id	Unique identifier for each user.
first_name	First name of user.
last_name	Last name of user.
email	Email will be used to verify the account and to log in to it.
phone_number	This will be used to contact the customer.
Encrypted_password	Unhashed passwords is sensitive so they need to be encrypted.

addresses

For organizational purposes, it is advisable to store the addresses of users and drivers in separate tables since they are different entities. However, in this particular case, both user and driver addresses are stored in the same table because they share the same fields. This approach is more memory-efficient while still maintaining an organized structure.

column	description
id	Unique identifier for each address.
house _number	House number.
street_name	Street name.
barangay	Barangay.
city	City.
postal_code	Postal codes could be used to calculate delivery charges.
region_id	Foreign key to regions look up table.

regions

This is a look up table for regions. This makes the design applicable to in national scale. It is decided to create a separate table because regions is a defined set of values.

column	description
id	Unique identifier for each region.
region_name	The specific regions of the Philippines (or really, any other country).

users_addresses

It is a common convention in Relational Database Management Systems to create a joining tables for entities that have many-to-many relationships. In this case, a user can have multiple addresses and multiple users can live in a single address.

column	description
user_id_	Foreign key referencing the users table.
address_id	Foreign key referencing the addresses table.

payment_methods

This table holds all the credit card information of a user. Some of these fields is going to be NULL for users with Cash On Delivery as their payment method.

column	description
id	Unique identifier for each region.
user_id_	Foreign key referencing the users table.
payment_type_id	Foreign key referencing the payment_types table.
bank	The bank who hosts the card.
expiry_date	The expiration date of the user's card.
is_default	Boolean value whether a payment method record is default for a user. Users can render multiple payment methods in their account.

payment_types

A way a transaction will be paid.

column	description
id	Unique identifier for each payment type.
payment_type	Refers to the method of payment used for a financial transaction.

products

These are the main products. If an app has an interface that displays the common categories first before proceeding to specific product items, this table will be used for displaying those main products.

column	description
id	Unique identifier for each product.
category_id	Foreign key referencing the product_categories. A product always has a category.
product_name	The display name for each product.
image_url	Link to product's image URL.

<dev>
</dev>



products_items

This represents a specific product, essentially transforming the abstract concept of 'products' into a tangible object.

column	description
id	Unique identifier for each product item.
product_id	Foreign key referencing the products table.
SKU	In ecommerce settings, SKUs (Stock Keeping Units) play a crucial role in managing inventory.
product_item_name	The name for the specific product item.
product_item_desc	The description for the specific product item.
product_image	The link to the specific product item's image URL.
price	Price of the item. This is numeric value taking into account the decimal places for cents.

products_categories

This is a self-referencing table to cater the hierarchy of categories. Instead of creating multiple tables off the same fields, it is a convention to use these kinds of tables. Example, a chocolate cake and a carrot cake sits under the cakes category but cakes category is also under the pastries category.

column	description
id	Unique identifier for each product category.
category	The categories available in your shop.
parent_category	Foreign key referencing the product_categories table, the same table.

variations

These are the qualities that changes for each category.

column	description
id	Unique identifier for each variation.
category_id	Foreign key referencing the product_categories table.
variation_type	The quality that changes, say size.

variation_values

The instances of each variation.

column	description
id	Unique identifier for each variation value.
variation_id	Foreign key referencing the variations table.
variation_value	The value of the variation type.

<dev>
</dev>



product_items_variation_values

This is another joining table. A certain product item could have many variations and variation values could be assigned to multiple product items.

column	description
product_item_id	Foreign key referencing the product_items table.
variation_value_id	Foreign key referencing the variation_values table.

product_inventory

This table contains basic information about the stocks of certain product item.

column	description
id	Unique identifier for each inventory item.
product_item_id	Foreign key referencing the product_items table.
qty_in_stock	Stocks left. This is set to numeric with 2 decimal places because most items in coffee shops are measured in grams and liters.
reorder_point	Serves as a guideline for determining when a specific product item should be replenished.
status	Status of stocks of a product item.

inventory_restock

This is created because restock dates are commonly fast-phase so we don't want to crowd the product_inventory table with redundant data. This is the reason why it is decided to have a separate table for these fields and just reference to this table.

column	description
id	Unique identifier for each restock record.
product_inventory_id	Foreign key referencing the product_items table.
last_restock_date	We need to track restocking times to gain insight into which product items are experiencing high turnover.
restock_qty	Refers to the number of units that have been replenished during the restocking process.

<dex>
</dev>



discount_offers

A shop holds seasonal discount offers.

column	description
id	Unique identifier for each discount offer.
product_inventory_id	Foreign key referencing the product_inventory table.
discount_name	Name of the discount.
discount_desc	Description of the discount.
discount_rate	Decimal value which will be used for calculations.
start_date_	Specifies the commencement date of the discount promotion, indicating when it becomes valid.
end_date_	Denotes the date on which the discount promotion concludes its validity.

discount_types

The types of discounts.

column	description
id	Unique identifier for each discount type.
discount_type	The type of discount offered.

discount_offers_categories

To manage discounts that are only applicable to specific products, we need to create a joining table.

column	description
discount_offer_id	Foreign key referencing the discount_offers table.
category_id	Foreign key referencing the product_categories table.

trays

Holds all the items a user selects. A user can have multiple trays. Each tray is for a different transaction. It is called tray to fit the coffee shop theme.

column	description
id	Unique identifier for each tray.
user_id_	Foreign key referencing the users table.

<dex>
</dev>



tray_items

A tray can hold multiple product items.

column	description
id	Unique identifier for each tray item.
tray_id	Foreign key referencing the trays table.
product_item_id	Foreign key referencing the product_items table.
qty	Quantity of each product item.
price	The price relative to the quantity.

order_transactions

Holds all information about a specific transaction.

column	description
id	Unique identifier for each order transaction.
user_id	Foreign key referencing the users table.
tray_id	Foreign key referencing the trays table.
payment_method_id	Foreign key referencing the payment_methods table.
address_id	Foreign key referencing the addresses table.
order_status_id	Foreign key referencing the order_status table.
order_date	Timestamp of order being placed.
order_total	Total order price of the transaction taking into account the multiple products in a tray.

order_status

Status of an order.

column	description
id	Unique identifier for each order status.
order_status	Values of order status.

user_reviews

A user can comment on their shopping experience.

column	description
id	Unique identifier for each user review.
user_id	Foreign key referencing the users table.
order_transaction_id	Foreign key referencing the order_transactions table.
rating_score	Customer rating, integer values from 1-5.
comment	Textual review of a customer.

<dex>
</dev>



delivery_drivers

Driver information. We need to care about which driver delivered to a customer.

column	description
id	Unique identifier for each driver.
first_name	First name of driver.
middle_name	Last name of driver.
last_name	Last name of driver.
email	Used to (secondary) contact the driver.
contact_number	Used to contact the driver.
birthday	To calculate age.
gender	Gender.
address_id	Foreign key referencing the addresses table.

drivers_license

Driver information. We need to care about which driver delivered to a customer.

column	description
id	Unique identifier for each driver.
delivery_driver_id	Foreign key referencing the delivery_drivers table.
license_number	The unique identification number associated with a driver's license.
license_expiration	The date when the driver's license is set to expire.
license_category	The category or type of license held by the driver (e.g., Class A, Class B).

vehicle_information

Details of the drivers vehicle.

column	description
id	Unique identifier for each driver.
delivery_driver_id	Foreign key referencing the delivery_drivers table.
vehicle_make	The manufacturer or brand of the vehicle.
vehicle_model	The specific model of the vehicle from the manufacturer.
vehicle_plate	The license plate number of the vehicle.

<dev>
</dev>



SQL SCRIPTS (for creating the database tables)

```
CREATE DATABASE coffee_ecommerce;
```

```
\c coffee_ecommerce
```

```
CREATE TABLE IF NOT EXISTS users(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    first_name VARCHAR(100) NOT NULL,  
    last_name VARCHAR(100) NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    phone_number VARCHAR(100) NOT NULL,  
    password_ VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS regions(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    region_name VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS addresses(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    house_number VARCHAR(20) NOT NULL,  
    street_name VARCHAR(100) NOT NULL,  
    barangay VARCHAR(100) NOT NULL,  
    city VARCHAR(100) NOT NULL,  
    postal_code VARCHAR(20) NOT NULL,  
    region_id BIGINT NOT NULL REFERENCES regions(id)  
);
```

<dex>
</dev>



```
CREATE TABLE IF NOT EXISTS users_addresses(  
    user_id_ BIGINT REFERENCES users(id),  
    address_id BIGINT REFERENCES addresses(id),  
    is_default BOOLEAN NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS payment_types(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    payment_type VARCHAR(100) NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS payment_methods(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    user_id_ BIGINT REFERENCES users(id),  
    payment_type_id BIGSERIAL REFERENCES payment_types(id),  
    bank VARCHAR(255),  
    account_number VARCHAR(100),  
    expiry_date_ DATE,  
    is_default BOOLEAN NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS product_categories(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    category VARCHAR(255) NOT NULL,  
    parent_category BIGINT REFERENCES product_categories(id)  
);
```

<dex>
</dev>



```
CREATE TABLE IF NOT EXISTS products(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    category_id BIGINT REFERENCES product_categories(id),  
    product_name VARCHAR(255) NOT NULL,  
    image_url VARCHAR(255) NOT NULL  
);
```


```
CREATE TABLE IF NOT EXISTS product_items(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    product_id BIGINT REFERENCES products(id),  
    SKU VARCHAR(255) NOT NULL,  
    product_item_name VARCHAR(255) NOT NULL,  
    product_item_desc VARCHAR(255) NOT NULL,  
    product_image VARCHAR(255) NOT NULL,  
    price NUMERIC(10, 2) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS variations(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    category_id BIGINT REFERENCES product_categories(id),  
    variation_type VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS variation_values(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    variation_id BIGINT REFERENCES variations(id),  
    variation_value VARCHAR(255) NOT NULL  
);
```

<dev>

</dev>




```
CREATE TABLE IF NOT EXISTS product_inventory(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    product_item_id BIGINT REFERENCES product_items(id),  
    qty_in_stock NUMERIC(10, 2),  
    reorder_point NUMERIC(10, 2) NOT NULL,  
    status VARCHAR(100)  
);
```

```
CREATE TABLE IF NOT EXISTS inventory_restock(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    product_inventory_id BIGINT REFERENCES product_inventory(id),  
    last_restock_date TIMESTAMP NOT NULL,  
    restock_qty NUMERIC(10,2) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS trays(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    user_id_ BIGINT NOT NULL REFERENCES users(id)  
);
```

```
CREATE TABLE IF NOT EXISTS tray_items(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    tray_id BIGINT REFERENCES trays(id),  
    product_item_id BIGINT REFERENCES product_items(id),  
    qty INT NOT NULL,  
    price NUMERIC(10, 2) NOT NULL  
);
```

<dex>
</dev>



```
CREATE TABLE IF NOT EXISTS order_status(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    order_status VARCHAR(100) NOT NULL  
);
```


```
CREATE TABLE IF NOT EXISTS order_transactions(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    user_id_ BIGINT REFERENCES users(id),  
    tray_id BIGINT REFERENCES trays(id),  
    payment_method_id BIGINT REFERENCES payment_methods(id),  
    address_id BIGINT REFERENCES addresses(id),  
    order_status BIGINT REFERENCES order_status(id),  
    order_date TIMESTAMP NOT NULL,  
    order_total NUMERIC(10, 2) NOT NULL  
);
```

```
CREATE TABLE IF NOT EXISTS user_reviews(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    user_id_ BIGINT REFERENCES users(id),  
    order_transaction_id BIGINT REFERENCES order_transactions(id),  
    rating_score INT,  
    comment TEXT  
);
```

```
CREATE TABLE IF NOT EXISTS discount_types(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    discount_type VARCHAR(255) NOT NULL  
);
```

<dev>

</dev>



```
CREATE TABLE IF NOT EXISTS discount_offers(  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    discount_type_id BIGINT REFERENCES discount_types(id),  
    discount_name VARCHAR(255) NOT NULL,  
    discount_desc VARCHAR(255) NOT NULL,  
    discount_rate NUMERIC(5, 2) NOT NULL,  
    start_date_ DATE NOT NULL,  
    end_date_ DATE NOT NULL  
);  
  
CREATE TABLE IF NOT EXISTS discount_offers_product_categories(  
    discount_offer_id BIGINT REFERENCES discount_offers(id),  
    category_id BIGINT REFERENCES product_categories(id)  
);  
  
CREATE TABLE IF NOT EXISTS delivery_drivers (  
    id BIGSERIAL PRIMARY KEY NOT NULL,  
    first_name VARCHAR(100) NOT NULL,  
    middle_name VARCHAR(100),  
    last_name VARCHAR(100) NOT NULL,  
    email VARCHAR(255),  
    contact_number VARCHAR(100) NOT NULL,  
    birthday DATE NOT NULL,  
    gender VARCHAR(10) NOT NULL,  
    address_id BIGINT REFERENCES addresses(id)  
);  
  
CREATE TABLE IF NOT EXISTS drivers_license (
```

<dev>

</dev>



```
id BIGSERIAL PRIMARY KEY NOT NULL,  
delivery_driver_id BIGINT REFERENCES delivery_drivers(id),  
license_number VARCHAR(100) NOT NULL,  
license_expiration_date DATE NOT NULL,  
license_category VARCHAR(100)  
);
```

```
CREATE TABLE IF NOT EXISTS vehicle_information (  
id BIGSERIAL PRIMARY KEY NOT NULL,  
delivery_driver_id BIGINT REFERENCES delivery_drivers(id),  
vehicle_make VARCHAR(100) NOT NULL,  
vehicle_model VARCHAR(100) NOT NULL,  
vehicle_plate VARCHAR(100) NOT NULL  
);
```


<dev>
</dev>



SAMPLE DATA (each table)

Refer to this link to view all the csv files of each and every table.

https://github.com/Dex-Astorga/coffee-shop-ecom/tree/main/csv_files

EXAMPLE QUERIES

Refer to this link to view all the csv files of each and every table.

https://github.com/Dex-Astorga/coffee-shop-ecom/tree/main/sample_queries

CONCLUSION

This database design is tailored to coffee shops, but these concepts can easily be applicable to other products like restaurants, bakeries, or retail stores. I really hope this one helps you. If you have any questions you can reach me at codex1727@gmail.com.

Thank you!

Dexter Astorga